```
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
```

```
VV      VV    AAAAAA    XX      XX  77777777      888888      000000    RRRRRRRR    EEEEEEEEEE    GGGGGGGG
VV      VV    AAAAAA    XX      XX  77777777      888888      000000    RRRRRRRR    EEEEEEEEEE    GGGGGGGG
VV      VV  AA      AA  XX      XX          77  88      88  00      00  RR      RR  EE          GG
VV      VV  AA      AA  XX      XX          77  88      88  00      00  RR      RR  EE          GG
VV      VV  AA      AA    XX  XX            77  88      88  00    0000  RR      RR  EE          GG
VV      VV  AA      AA    XX  XX            77  88      88  00    0000  RR      RR  EE          GG
VV      VV  AA      AA      XX          77      888888    00  00  00  RRRRRRRR    EEEEEEEE    GG
VV      VV  AA      AA      XX          77      888888    00  00  00  RRRRRRRR    EEEEEEEE    GG
VV      VV  AAAAAAAAAA    XX  XX        77      88      88  0000    00  RR  RR      EE          GG  GGGGGG
VV      VV  AAAAAAAAAA    XX  XX        77      88      88  0000    00  RR  RR      EE          GG  GGGGGG
  VV  VV    AA      AA  XX      XX    77        88      88  00      00  RR    RR    EE          GG      GG    ....
  VV  VV    AA      AA  XX      XX    77        88      88  00      00  RR    RR    EE          GG      GG    ....
    VV      AA      AA  XX      XX  77            888888      000000    RR      RR  EEEEEEEEEE    GGGGGG      ....
    VV      AA      AA  XX      XX  77            888888      000000    RR      RR  EEEEEEEEEE    GGGGGG      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
C
C Version:      'V04-000'
C
C*************************************************************************
C*                                                                       *
C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.               *
C*  ALL RIGHTS RESERVED.                                                 *
C*                                                                       *
C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE *
C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER *
C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY *
C*  TRANSFERRED.                                                         *
C*                                                                       *
C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE *
C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT *
C*  CORPORATION.                                                         *
C*                                                                       *
C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS *
C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.              *
C*                                                                       *
C*                                                                       *
C*************************************************************************
C

        SUBROUTINE VAX780REG (LUN,REGISTER)

C
C       AUTHOR  BRIAN PORTER                CREATION DATE   9-AUG-1979
C
C       Functional desciption:
C
C       This module displays all the Kernel registers that appear in various
C       error log entries for the 11/780.
C
C       Modified by:
C
C       V03-006 SAR0201         Sharon A. Reynolds,     27-Feb-1984
C               Added an SYE update that fixes a bug in the
C               SBI confirmation codes in SBIERR.
C
C       V03-005 SAR0162         Sharon A. Reynolds,     13-Oct-1983
C               Added an SYE update that adds 11/7XX support.
C
C       V03-004 SAR0106         Sharon A. Reynolds,     20-Jun-1983
C               Changed the carriage control in the 'format' statements
C               for use with ERF.
C
C       v03-003 BP0012          Brian Porter,           27-FEB-1983
C               Enhanced sbi silo output.
C
C       v03-002 BP0011          Brian Porter,           19-AUG-1982
C               Corrected ms780e memory size.
C
C       v03-001 BP0010          Brian Porter,           19-FEB-1982
```

```
C                       Added ci780 support.
C**


        BYTE            LUN

        INTEGER*4       REGISTER

        INTEGER*4       FIELD

        INTEGER*4       COMPRESSC

        integer*4       compress4

        integer*4       lib$extzv

        INTEGER*4       FIELD1

        PARAMETER       FOUR_K = 1

        PARAMETER       SIXTEEN_K = 2

        PARAMETER       MA780_0 = 64

        PARAMETER       MA780_3 = 67

        PARAMETER       UBA_0 = 40

        PARAMETER       UBA_3 = 43

        PARAMETER       MS780_4K = 8

        PARAMETER       MS780_4KI = 9

        PARAMETER       MS780_16K = 16

        PARAMETER       MS780_16KI = 17

        PARAMETER       DR780 = 48

        PARAMETER       MBA = 32

        PARAMETER       COMMAND_ADDRESS = 3

        PARAMETER       READ_DATA = 0

        CHARACTER*21    V1UBA_REGA(16:18)

        CHARACTER*17    V1MS780C_REGA(0:0)

        character*17    v1ms780e_rega(8:8)

        equivalence     (v1ms780c_rega,v1ms780e_rega)

        character*30    v2ms780e_rega(15:20)

        CHARACTER*26    MS780C_RAM_TYPE(0:3)

        character*26    ms780e_ram_type(0:3)

        character*31    ms780e_interleave_mode(0:4)

        CHARACTER*24    V2DRCR(11:11)

        CHARACTER*21    V3DRCR(15:20)

        CHARACTER*15    V4DRCR(24:24)

        CHARACTER*25    V1DRCR(1:3)

        CHARACTER*12    SBI_CONFIRM(1:3)

        CHARACTER*22    V1SBI_ERROR(1:3)

        CHARACTER*23    V2SBI_ERROR(7:8)

        CHARACTER*25    V3SBI_ERROR(13:15)

        CHARACTER*29    V1TIMEOUT_ADDR(29:29)

        CHARACTER*15    ACCS_TYPE(0:2)

        CHARACTER*20    V1ACCS(15:15)

        CHARACTER*23    V2ACCS(27:29)

        CHARACTER*6     V3ACCS(31:31)

        CHARACTER*23    V1SBI_FAULT(16:19)

        CHARACTER*31    V2SBI_FAULT(26:31)

        CHARACTER*31    V1SBI_REGA(21:23)

        CHARACTER*31    V2SBI_REGA(26:31)

        CHARACTER*27    V1SBI_COMPARATR(29:31)

        CHARACTER*22    TIMEOUT_STATUS(0:3)

        CHARACTER*22    IB_STATUS(0:3)

        CHARACTER*22    CP_STATUS(0:3)

        CHARACTER*11    REF_MODE(0:3)

        CHARACTER*21    SBI_RESPONSE(0:2)

        CHARACTER*17    V1SBI_SILO(30:31)

        CHARACTER*18    SBI_TAG(0:7)

        CHARACTER*25    COND_LOCK(1:3)

        CHARACTER*23    SBI_FUNCTION(0:12)

        character*19    v1ci780_rega(8:10)

        character*25    v2ci780_rega(16:20)

        EQUIVALENCE     (V2SBI_REGA,V2SBI_FAULT)

        EQUIVALENCE     (TIMEOUT_STATUS,IB_STATUS)

        EQUIVALENCE     (TIMEOUT_STATUS,CP_STATUS)



        DATA    V1UBA_REGA(16)  /'UNIBUS INIT COMPLETE*'/

        DATA    V1UBA_REGA(17)  /'UNIBUS POWER DOWN*'/

        DATA    V1UBA_REGA(18)  /'UNIBUS INIT ASSERTED*'/



        DATA    MS780C_RAM_TYPE(0)      /'NO ARRAY BOARDS PRESENT*'/

        DATA    MS780C_RAM_TYPE(1)      /'4K RAM ARRAY BOARDS*'/

        DATA    MS780C_RAM_TYPE(2)      /'16K RAM ARRAY BOARDS*'/

        DATA    MS780C_RAM_TYPE(3)      /'MULTIPLE ARRAY TYPE ERROR*'/



        data    ms780e_ram_type(0)      /'MULTIPLE ARRAY TYPE ERROR*'/

        data    ms780e_ram_type(1)      /'64K RAM ARRAY BOARDS*'/

        data    ms780e_ram_type(2)      /'256K RAM ARRAY BOARDS*'/

        data    ms780e_ram_type(3)      /'NO ARRAY BOARDS PRESENT*'/




        DATA    V1MS780C_REGA(0)/'INTERLEAVED MODE*'/





        data    v2ms780e_rega(15)       /'LOWER MISCONFIGURATION*'/

        data    v2ms780e_rega(16)       /'UPPER MISCONFIGURATION*'/

        data    v2ms780e_rega(17)       /'INTERLEAVE MISCONFIGURATION*'/

        data    v2ms780e_rega(18)       /'LOWER CONTROLLER PARITY ERROR*'/

        data    v2ms780e_rega(19)       /'UPPER CONTROLLER PARITY ERROR*'/

        data    v2ms780e_rega(20)       /'ERROR SUMMARY*'/




        data    ms780e_interleave_mode(0)
        1 /'NON-INTERLEAVED (LOWER)*'/

        data    ms780e_interleave_mode(1)
        1 /'EXTERNALLY INTERLEAVE (LOWER)*'/

        data    ms780e_interleave_mode(2)
        1 /'NON-INTERLEAVED (UPPER)*'/

        data    ms780e_interleave_mode(3)
        1 /'EXTERNALLY INTERLEAVED (UPPER)*'/

        data    ms780e_interleave_mode(4)
        1 /'INTERNALLY 2-WAY INTERLEAVED*'/





        DATA    V1DRCR(1)       /'INTERLOCK SEQUENCE FAULT*'/

        DATA    V1DRCR(2)       /'READ DATA TIMEOUT FAULT*'/

        DATA    V1DRCR(3)       /'ILLEGAL TIMEOUT STATUS*'/



        DATA    V2DRCR(11)      /'DDI DATA STALL*'/



        DATA    V3DRCR(15)      /'READ DATA SUBSTITUTE*'/

        DATA    V3DRCR(16)      /'CORRECTED READ DATA*'/

        DATA    V3DRCR(17)      /'MICRO-CODE HALTED*'/

        DATA    V3DRCR(18)      /'ABORT*'/

        DATA    V3DRCR(19)      /'PACKET INTERRUPT*'/

        DATA    V3DRCR(20)      /'INTERRUPT ENABLE*'/



        DATA    V4DRCR(24)      /'EXTERNAL ABORT*'/



        DATA    SBI_CONFIRM(1)  /'ACKNOWLEDGE*'/

        DATA    SBI_CONFIRM(2)  /'BUSY*'/

        DATA    SBI_CONFIRM(3)  /'ERROR*'/



        DATA    V1SBI_ERROR(1)  /'SBI NOT BUSY*'/

        DATA    V1SBI_ERROR(2)  /'MULTIPLE CPU ERROR*'/

        DATA    V1SBI_ERROR(3)  /'IB ERROR CONFIRMATION*'/




        DATA    V2SBI_ERROR(7)  /'IB RECEIVED RDS*'/

        DATA    V2SBI_ERROR(8)  /'CPU ERROR CONFIRMATION*'/



        DATA    V3SBI_ERROR(13) /'RDS CONFIRMATION*'/

        DATA    V3SBI_ERROR(14) /'CRD CONFIRMATION*'/

        DATA    V3SBI_ERROR(15) /'RDS/CRD INTERRUPT ENABLE*'/






        DATA    V1TIMEOUT_ADDR  /'PROTECTION CHECKED REFERENCE*'/



        DATA    ACCS_TYPE(0)    /'NOT PRESENT*'/

        DATA    ACCS_TYPE(1)    /'FLOATING POINT*'/

        DATA    ACCS_TYPE(2)    /'UNKNOWN*'/



        DATA    V1ACCS(15)      /'ACCELERATOR ENABLED*'/



        DATA    V2ACCS(27)      /'RESERVED OPERAND ERROR*'/

        DATA    V2ACCS(28)      /'OVERFLOW ERROR*'/

        DATA    V2ACCS(29)      /'UNDERFLOW ERROR*'/



        DATA    V3ACCS(31)      /'ERROR*'/




        DATA    V1SBI_FAULT(16) /'FAULT SILO LOCK*'/

        DATA    V1SBI_FAULT(17) /'SBI FAULT*'/

        DATA    V1SBI_FAULT(18) /'FAULT INTERRUPT ENABLE*'/

        DATA    V1SBI_FAULT(19) /'FAULT LATCH*'/





        DATA    V1SBI_REGA(21)/'ADAPTER OVER-TEMPERATURE*'/

        DATA    V1SBI_REGA(22)/'ADAPTER POWER-UP*'/

        DATA    V1SBI_REGA(23)/'ADAPTER POWER-DOWN*'/




        DATA    V2SBI_REGA(26)/'TRANSMITTER DURING FAULT CYCLE*'/

        DATA    V2SBI_REGA(27)/'MULTIPLE TRANSMITTER FAULT*'/

        DATA    V2SBI_REGA(28)/'INTERLOCK SEQUENCE FAULT*'/

        DATA    V2SBI_REGA(29)/'UNEXPECTED READ DATA FAULT*'/

        DATA    V2SBI_REGA(30)/'WRITE SEQUENCE FAULT*'/

        DATA    V2SBI_REGA(31)/'PARITY FAULT*'/




        DATA    V1SBI_COMPARATR(29)/'LOCK UNCONDITIONAL*'/

        DATA    V1SBI_COMPARATR(30)/'SILO LOCK INTERRUPT ENABLE*'/

        DATA    V1SBI_COMPARATR(31)/'COMPARATOR SILO LOCK*'/



        DATA    TIMEOUT_STATUS(0)/'NO RESPONSE*'/

        DATA    TIMEOUT_STATUS(1)/'DEVICE BUSY*'/

        DATA    TIMEOUT_STATUS(2)/'WAITING FOR READ DATA*'/

        DATA    TIMEOUT_STATUS(3)/'ILLEGAL*'/


```

VAX

```
	DATA	REF_MODE(0)	/'KERNEL*'/

	DATA	REF_MODE(1)	/'EXECUTIVE*'/

	DATA	REF_MODE(2)	/'SUPERVISOR*'/

	DATA	REF_MODE(3)	/'USER*'/




	DATA	SBI_RESPONSE(0) /'ERROR FREE DATA*'/

	DATA	SBI_RESPONSE(1) /'CORRECTED READ DATA*'/

	DATA	SBI_RESPONSE(2) /'READ DATA SUBSTITUTE*'/





	DATA	V1SBI_SILO(30)	/'SBI INTERLOCKED*'/

	DATA	V1SBI_SILO(31)	/'FAULT CLEAR FLAG*'/




	DATA	SBI_TAG(0)	/'READ DATA*'/

	DATA	SBI_TAG(1)	/'ILLEGAL TAG*'/

	DATA	SBI_TAG(2)	/'ILLEGAL TAG*'/

	DATA	SBI_TAG(3)	/'COMMAND ADDRESS*'/

	DATA	SBI_TAG(4)	/'ILLEGAL TAG*'/

	DATA	SBI_TAG(5)	/'WRITE DATA*'/

	DATA	SBI_TAG(6)	/'INTERRUPT SUMMARY*'/

	DATA	SBI_TAG(7)	/'ILLEGAL TAG*'/




	DATA	COND_LOCK(1)	/'ID ONLY*'/

	DATA	COND_LOCK(2)	/'ID AND TAG*'/

	DATA	COND_LOCK(3)	/'ID, TAG AND COMMAND/MASK*'/
```

```

        DATA    SBI_FUNCTION(0) /'ILLEGAL FUNCTION*'/

        DATA    SBI_FUNCTION(1) /'READ MASKED*'/

        DATA    SBI_FUNCTION(2) /'WRITE MASKED*'/

        DATA    SBI_FUNCTION(3) /'ILLEGAL FUNCTION*'/

        DATA    SBI_FUNCTION(4) /'INTERLOCK READ MASKED*'/

        DATA    SBI_FUNCTION(5) /'ILLEGAL FUNCTION*'/

        DATA    SBI_FUNCTION(6) /'ILLEGAL FUNCTION*'/

        DATA    SBI_FUNCTION(7) /'INTERLOCK WRITE MASKED*'/

        DATA    SBI_FUNCTION(8) /'EXTENDED READ*'/

        DATA    SBI_FUNCTION(9) /'ILLEGAL FUNCTION*'/

        DATA    SBI_FUNCTION(10)/'ILLEGAL FUNCTION*'/

        DATA    SBI_FUNCTION(11)/'EXTENDED WRITE MASKED*'/

        DATA    SBI_FUNCTION(12)/'ILLEGAL FUNCTION*'/



        data    v1ci780_rega(8) /'POWER-FAIL DISABLE*'/

        data    v1ci780_rega(9) /'TRANSMIT DEAD*'/

        data    v1ci780_rega(10)/'TRANSMIT FAIL*'/




        data    v2ci780_rega(16)/'CORRECTED READ DATA*'/

        data    v2ci780_rega(17)/'READ DATA SUBSTITUTE*'/

        data    v2ci780_rega(18)/'COMMAND TRANSMIT ERROR*'/

        data    v2ci780_rega(19)/'READ DATA TIMEOUT*'/

        data    v2ci780_rega(20)/'COMMAND TRANSMIT TIMEOUT*'/




        ENTRY ACCS_780 (LUN,REGISTER)


        FIELD = LIB$EXTZV(0,8,REGISTER)

        CALL LINCHK (LUN,2)

        WRITE(LUN,10) REGISTER,ACCS_TYPE(MIN(2,FIELD))
10      FORMAT(' ',T8,'ACCS',T24,Z8.8,/,T40,'ACCELERATOR ',
        1 A<COMPRESSC (ACCS_TYPE(MIN(2,FIELD)))>)

        IF (FIELD .EQ. 1
        1 .OR.
        2 FIELD .EQ. 2) THEN

        CALL OUTPUT (LUN,REGISTER,V1ACCS,15,15,15,'0')

        CALL OUTPUT (LUN,REGISTER,V2ACCS,27,27,29,'0')

        CALL OUTPUT (LUN,REGISTER,V3ACCS,31,31,31,'0')
        ENDIF

        RETURN



        ENTRY SBI_FAULTREG (LUN,REGISTER)


        CALL LINCHK (LUN,1)

        WRITE(LUN,30) REGISTER
30      FORMAT(' ',T8,'SBIFS',T24,Z8.8)

        CALL OUTPUT (LUN,REGISTER,V1SBI_FAULT,16,16,19,'0')

        CALL OUTPUT (LUN,REGISTER,V2SBI_FAULT,26,26,31,'0')

        RETURN



        ENTRY SBI_COMPARATOR (LUN,REGISTER)


        CALL LINCHK (LUN,1)

        WRITE(LUN,40) REGISTER
40      FORMAT(' ',T8,'SBISC',T24,Z8.8)

        FIELD = LIB$EXTZV(16,4,REGISTER)

        IF (FIELD .NE. 0) THEN

        CALL LINCHK (LUN,1)

        WRITE(LUN,50) FIELD
50      FORMAT(' ',T40,'COUNT FIELD = ',I2,'.')
        ENDIF

        FIELD = LIB$EXTZV(20,3,REGISTER)

        IF (FIELD .NE. 0) THEN
```

```

          CALL LINCHK (LUN,1)

          WRITE(LUN,60) SBI_TAG(FIELD)
 60       FORMAT(' ',T40,'COMPARE TAG = ',A<COMPRESSC (SBI_TAG(FIELD))>)
          ENDIF

          FIELD = LIB$EXTZV(23,4,REGISTER)

          IF (FIELD .NE. 0) THEN

          CALL LINCHK (LUN,1)

          WRITE(LUN,70) FIELD
 70       FORMAT(' ',T40,'COMPARE COMMAND/MASK = ',I2,'.')
          ENDIF

          FIELD = LIB$EXTZV(27,2,REGISTER)

          IF (FIELD .NE. 0) THEN

          CALL LINCHK (LUN,1)

          WRITE(LUN,80) COND_LOCK(FIELD)
 80       FORMAT(' ',T40,'LOCK = ',A<COMPRESSC (COND_LOCK(FIELD))>)
          ENDIF

          CALL OUTPUT (LUN,REGISTER,V1SBI_COMPARATR,29,29,31,'0')

          RETURN



          ENTRY SBI_MAINTENANCE (LUN,REGISTER)


          CALL LINCHK (LUN,1)

          WRITE(LUN,90) REGISTER
 90       FORMAT(' ',T8,'SBIMT',T24,Z8.8)

          IF (JIAND(REGISTER,'F05FF900'X) .NE. 0) THEN

          CALL LINCHK (LUN,1)

          WRITE(LUN,100)
 100      FORMAT(' ',T40,'DIAGNOSTIC MODE')
          ELSE

          IF (JIAND(REGISTER,'200000'X) .EQ. 0) THEN

          CALL LINCHK (LUN,1)

          WRITE(LUN,105)
 105      FORMAT(' ',T40,'SBI INVALIDATE DISABLED')
          ENDIF
          ENDIF
```

```

      RETURN



      ENTRY SBI_ERROR (LUN,REGISTER)



      CALL LINCHK (LUN,1)

      WRITE(LUN,135) REGISTER
135   FORMAT(' ',T8,'SBIER',T24,Z8.8)

      CALL OUTPUT (LUN,REGISTER,V1SBI_ERROR,1,1,3,'0')

      FIELD = LIB$EXTZV(6,1,REGISTER)

      IF (FIELD .NE. 0) THEN

      FIELD = LIB$EXTZV(4,2,REGISTER)

      CALL LINCHK (LUN,2)

      WRITE(LUN,140) IB_STATUS(FIELD)
140   FORMAT(' ',T40,5H IB ',A<COMPRESSC (IB_STATUS(FIELD))>,' TIMEOUT',
     1 T40,'IB TIMEOUT')
      ENDIF

      CALL OUTPUT (LUN,REGISTER,V2SBI_ERROR,7,7,8,'0')

      FIELD = LIB$EXTZV(12,1,REGISTER)

      IF (FIELD .NE. 0) THEN

      Field = LIB$EXTZV(10,2,register)

      CALL LINCHK (LUN,2)

      WRITE(LUN,150) CP_STATUS(FIELD)
150   FORMAT(' ',T40,A<COMPRESSC (CP_STATUS(FIELD))>,
     1 ' TIMEOUT',/,T40,'CPU TIMEOUT')
      ENDIF

      CALL OUTPUT (LUN,REGISTER,V3SBI_ERROR,13,13,15,'0')

      RETURN



      ENTRY SBI_TIMEOUT (LUN,REGISTER)



      CALL LINCHK (LUN,1)

      WRITE(LUN,155) REGISTER
```

```
155     FORMAT(' ',T8,'SBITA',T24,Z8.8)

        FIELD = LIB$EXTZV(0,28,REGISTER)

        CALL LINCHK (LUN,1)

        WRITE(LUN,160) JISHFT(FIELD,2)
160     FORMAT(' ',T40,'TIMEOUT CONSOLE ADDR = ',Z8.8)

        CALL OUTPUT (LUN,REGISTER,V1TIMEOUT_ADDR,29,29,29,'0')

        FIELD = LIB$EXTZV(30,2,REGISTER)

        CALL LINCHK (LUN,1)

        WRITE(LUN,170) REF_MODE(FIELD)
170     FORMAT(' ',T40,'TIMEOUT REFERENCE IN ',
        1 A<COMPRESSC (REF_MODE(FIELD))>,' MODE')

        RETURN



        ENTRY SBI_SILO (LUN,REGISTER)



        CALL LINCHK (LUN,1)

        WRITE(LUN,175) REGISTER
175     FORMAT(' ',T24,Z8.8)

        DO 183,J = 0,15

        FIELD = LIB$EXTZV(J,1,REGISTER)

        IF (FIELD .NE. 0) THEN

        CALL LINCHK (LUN,1)

        WRITE(LUN,180) J
180     FORMAT(' ',T40,'TR ',I2.2,'. ACTIVE')
        ENDIF

183     CONTINUE

        FIELD = LIB$EXTZV(16,2,REGISTER)

        if (
        1 field .ge. 1
        1 .and.
        1 field .le. 3
        1 ) then

        CALL LINCHK (LUN,1)

        WRITE(LUN,185) SBI_CONFIRM(FIELD)
```

```
185     FORMAT(' ',T40,'CONFIRMATION = ',
        1 A<COMPRESSC (SBI_CONFIRM(FIELD))>)
        endif

        if (lib$extzv(18,12,register) .ne. 0) then

        FIELD1 = LIB$EXTZV(22,3,REGISTER)

        FIELD = LIB$EXTZV(18,4,REGISTER)

        CALL LINCHK (LUN,1)

        IF (FIELD1 .EQ. COMMAND_ADDRESS) THEN

        WRITE(LUN,187) SBI_FUNCTION(MIN(12,FIELD))
187     FORMAT(' ',T40,'FUNCTION = ',
        1 A<COMPRESSC (SBI_FUNCTION(MIN(12,FIELD)))>)

        ELSE IF (FIELD1 .EQ. READ_DATA) THEN

        WRITE(LUN,189) SBI_RESPONSE(MIN(2,FIELD))
189     FORMAT(' ',T40,'DATA READ = ',
        1 A<COMPRESSC (SBI_RESPONSE(MIN(2,FIELD)))>)
        ELSE

        WRITE(LUN,191) FIELD
191     FORMAT(' ',T40,'MASK = ',Z1)
        ENDIF

        FIELD = LIB$EXTZV(22,3,REGISTER)

        CALL LINCHK (LUN,1)

        WRITE(LUN,193) SBI_TAG(FIELD)
193     FORMAT(' ',T40,'TAG = ',A<COMPRESSC (SBI_TAG(FIELD))>)

        FIELD = LIB$EXTZV(25,5,REGISTER)

        CALL LINCHK (LUN,1)

        if (field .ne. 16) then

        WRITE(LUN,197) FIELD
197     FORMAT(' ',T40,'ID  = ',I2.2)
        else

        write(lun,198)
198     format(' ',t40,'ID = CPU')
        endif
        endif

        CALL OUTPUT (LUN,REGISTER,V1SBI_SILO,30,30,31,'0')

        RETURN
```

```
VAX

PRO

0
1
2
3

ENT

0-

VAR

AP
```

```
                ENTRY SBI_COMMAND (LUN,REGISTER)


                CALL LINCHK (LUN,1)

                WRITE(LUN,200) SBI_FUNCTION(MIN(12,REGISTER))
        200     FORMAT(' ',T40,'FUNCTION = ',
                1 A<COMPRESSC (SBI_FUNCTION(MIN(12,REGISTER)))>)

                RETURN




                ENTRY DR780_REGA (LUN,REGISTER)




                CALL LINCHK (LUN,2)

                WRITE(LUN,209) REGISTER
        209     FORMAT(/' ',T8,'"DR" CR',T24,Z8.8)

                FIELD = LIB$EXTZV(0,8,REGISTER)

                CALL LINCHK (LUN,1)

                IF (FIELD .NE. DR780) THEN

                WRITE(LUN,210)
        210     FORMAT(' ',T40,'ADAPTER NOT "DR"')
                ELSE

                WRITE(LUN,215)
        215     FORMAT(' ',T40,'ADAPTER IS "DR"')

                DO 230,J = 2,3

                DO 230,I = J*4,(J*4) + 3

                FIELD = LIB$EXTZV(I,3,REGISTER)

                IF ((FIELD*2)/2 .NE. FIELD) THEN

                CALL LINCHK (LUN,2)

                WRITE(LUN,220) (J-1),V1DRCR(MIN(3,(FIELD+1)/2)),(J-1)
        220     FORMAT(' ',T40,'ID ',I1,'. ERROR',/,
                1 T40,A<COMPRESSC (V1DRCR(MIN(3,(FIELD+1)/2)))>,' ID ',I1,'.')
                ENDIF

                if (
                1 j .eq. 2
                1 .and.
                1 i .eq. 8
                1 ) then
```

```
VAX
ARR
3
2
2
2
LAB
1
FUN
T
COM
F
/
/
/
/
/
COM
R
E
P
D
```

```
                call output (lun,register,v2drcr,11,11,11,'0')
                endif

        230     CONTINUE

                CALL OUTPUT (LUN,REGISTER,V3DRCR,15,15,20,'0')

                CALL OUTPUT (LUN,REGISTER,V1SBI_REGA,21,21,23,'0')

                CALL OUTPUT (LUN,REGISTER,V4DRCR,24,24,24,'0')

                CALL OUTPUT (LUN,REGISTER,V2SBI_REGA,26,26,31,'0')
                ENDIF

                RETURN




                ENTRY MS780C_REGA (LUN,REGISTER)



                CALL LINCHK (LUN,2)

                WRITE(LUN,240) REGISTER
        240     FORMAT(/' ',T8,'CSRA',T24,Z8.8)

                if (lib$extzv (5,3,register) .ne. 0) then

                call linchk (lun,1)

                WRITE(LUN,250)
        250     FORMAT(' ',T40,'ADAPTER NOT MEMORY TYPE "C"')
                else

                CALL OUTPUT (LUN,REGISTER,V1MS780C_REGA,0,0,0,'0')

                FIELD = LIB$EXTZV(3,2,REGISTER)

                CALL LINCHK (LUN,1)

                WRITE(LUN,256) MS780C_RAM_TYPE(FIELD)
        256     FORMAT(' ',T40,A<COMPRESSC (MS780C_RAM_TYPE(FIELD))>)

                IF (FIELD .NE. 0) THEN

                IF (FIELD .EQ. FOUR_K) THEN

                FIELD = LIB$EXTZV(9,4,REGISTER)

                ELSE IF (FIELD .EQ. SIXTEEN_K) THEN

                FIELD = LIB$EXTZV(9,6,REGISTER)
                ENDIF
```

**F

```
                field = (field+1)*64

                call linchk (lun,1)

                WRITE(LUN,255)
        255     FORMAT(' ',T40,'ADAPTER IS MEMORY TYPE "C"')

                CALL LINCHK (LUN,1)

                WRITE(LUN,260) FIELD
        260     FORMAT(' ',T40,'MEMORY SIZE = ',i<compress4 (field)>,'.K')
                ENDIF

                CALL OUTPUT (LUN,REGISTER,V1SBI_REGA,21,22,23,'0')

                CALL OUTPUT (LUN,REGISTER,V2SBI_REGA,26,26,28,'0')

                call output (lun,register,v2sbi_rega,26,30,31,'0')
                endif

                RETURN




                entry ms780e_rega (lun,register)




                call linchk (lun,2)

                write(lun,261) register
        261     format(/' ',t8,'CSRA',t24,z8.8)

                if (lib$extzv(5,3,register) .ne. 3) then

                call linchk (lun,1)

                write(lun,262) 'ADAPTER NOT MEMORY TYPE "E"'
        262     format(' ',t40,a)
                else

                field = lib$extzv(0,3,register)

                if (field .le. 4) then

                call linchk (lun,1)

                write(lun,263) ms780e_interleave_mode(field)
        263     format(' ',t40,a<compressc (ms780e_interleave_mode(field))>)
                endif

                field = lib$extzv(3,2,register)

                call linchk (lun,1)
```

```
              write(lun,264) ms780e_ram_type(field)
      264     format(' ',t40,a<compressc (ms780e_ram_type(field))>)

              call linchk (lun,1)

              write(lun,262) 'ADAPTER IS MEMORY TYPE "E"'

              call output (lun,register,v1ms780e_rega,8,8,8,'0')

              field = lib$extzv(9,6,register) + 1

              call linchk (lun,1)

              write(lun,265) field
      265     format(' ',t40,'MEMORY SIZE = ',i<compress4 (field)>,'.M')

              call output (lun,register,v2ms780e_rega,15,15,20,'0')

              call output (lun,register,v1sbi_rega,21,22,23,'0')

              call output (lun,register,v2sbi_rega,26,26,28,'0')

              call output (lun,register,v2sbi_rega,26,30,31,'0')
              endif

              return




              ENTRY MA780_REGA (LUN,REGISTER)




              CALL LINCHK (LUN,2)

              WRITE(LUN,269) REGISTER
      269     FORMAT(/' ',T8,'PRTCFNG',T24,Z8.8)

              FIELD = LIB$EXTZV(0,8,REGISTER)

              IF (FIELD .LT. MA780_0 .OR. FIELD .GT. MA780_3) THEN

              CALL LINCHK (LUN,1)

              WRITE(LUN,275)
      275     FORMAT(' ',T40,'ADAPTER NOT MULTI-PORT MEMORY')
              ELSE

              FIELD = LIB$EXTZV(0,2,REGISTER)

              CALL LINCHK (LUN,2)

              WRITE(LUN,277) FIELD
      277     FORMAT(' ',T40,'ADAPTER IS MULTI-PORT MEMORY',/,
              1 T40,'PORT NUMBER = ',I1,'.')

              CALL OUTPUT (LUN,REGISTER,V1SBI_REGA,21,21,23,'0')

              CALL OUTPUT (LUN,REGISTER,V2SBI_REGA,26,26,31,'0')
              ENDIF

              RETURN




              entry rh780_configuration_register (lun,register)




              call linchk (lun,2)

              write(lun,279) register
      279     format(/' ',t8,'''RH'' CSR',t24,z8.8)

              field = lib$extzv(0,8,register)

              call linchk (lun,1)

              if (field .ne. mba) then

              write(lun,280)
      280     format(' ',t40,'ADAPTER NOT MBA')
              else

              write(lun,285)
      285     format(' ',t40,'ADAPTER IS MBA')

              call output (lun,register,v1sbi_rega,21,21,23,'0')

              call output (lun,register,v2sbi_rega,26,26,31,'0')
              endif

              return




              ENTRY UBA_REGA (LUN,REGISTER)




              CALL LINCHK (LUN,2)

              WRITE(LUN,289) REGISTER
      289     FORMAT(/' ',T8,'''DW'' CSR',T24,Z8.8)

              FIELD = LIB$EXTZV(0,8,REGISTER)

              CALL LINCHK (LUN,1)
```

```
VEC

011
011
011
011
011
012
012
012
012
012
012
012
012
012
012


PRO

  0
  1
  2


ENT

  0

VAR

  2

ARR
```

```

        IF (FIELD .LT. UBA_0 .OR. FIELD .GT. UBA_3) THEN

        WRITE(LUN,295)
295     FORMAT(' ',T40,'ADAPTER NOT UBA')
        ELSE

        FIELD = LIB$EXTZV(0,2,REGISTER)

        WRITE(LUN,300) FIELD
300     FORMAT(/' ',T40,'ADAPTER IS UBA ',I1,'.')

        CALL OUTPUT (LUN,REGISTER,V1UBA_REGA,16,16,18,'0')

        CALL OUTPUT (LUN,REGISTER,V1SBI_REGA,21,21,23,'0')

        CALL OUTPUT (LUN,REGISTER,V2SBI_REGA,26,26,31,'0')
        ENDIF

        RETURN




        entry ci780_rega (lun,register)




        call linchk (lun,2)

        write(lun,400) register
400     format(/' ',t8,'CNFGR',t24,z8.8)

        call linchk (lun,1)

        if (lib$extzv(0,8,register) .ne. '38'x) then

        write(lun,405) 'ADAPTER NOT "CI"'
405     format(' ',t40,a)
        else

        write(lun,405) 'ADAPTER IS "CI"'

        call output (lun,register,v1ci780_rega,8,8,10,'0')

        call output (lun,register,v2ci780_rega,16,16,20,'0')

        call output (lun,register,v1sbi_rega,21,22,23,'0')

        call output (lun,register,v2sbi_rega,26,26,27,'0')

        call output (lun,register,v2sbi_rega,26,29,31,'0')
        endif

        return

```

```
1141            END



PROGRAM SECTIONS

    Name                                        Bytes   Attributes

  0 $CODE                                        5228   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                       1474   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                       5168   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG

    Total Space Allocated                       11870

ENTRY POINTS

    Address   Type  Name                                Address   Type  Name

  0-00000010        ACCS_780                          0-000010F7        CI780_REGA
  0-000008BA        DR780_REGA                        0-00000E45        MA780_REGA
  0-00000A73        MS780C_REGA                       0-00000C2F        MS780E_REGA
  0-00000F3E        RH780_CONFIGURATION_REGISTER      0-0000085C        SBI_COMMAND
  0-0000013B        SBI_COMPARATOR                    0-0000036A        SBI_ERROR
  0-000000D1        SBI_FAULTREG                      0-000002C6        SBI_MAINTENANCE
  0-000005B0        SBI_SILO                          0-000004BB        SBI_TIMEOUT
  0-00001000        UBA_REGA                          0-00000000        VAX780REG

VARIABLES

    Address   Type  Name                Address   Type  Name                Address   Type  Name                Address   Type  Name

  2-00000A4C  I*4   FIELD             2-00000A50  I*4   FIELD1            2-00000A58  I*4   I                 2-00000A54  I*4   J
 AP-00000004@ L*1   LUN               2-00000A5C@ I*4   REGISTER

ARRAYS

    Address   Type  Name                                     Bytes  Dimensions

  2-0000056D  CHAR ACCS_TYPE                                    45  (0:2)
  2-00000820  CHAR COND_LOCK                                    75  (3)
  2-00000000  CHAR CP_STATUS                                    88  (0:3)
  2-00000000  CHAR IB_STATUS                                    88  (0:3)
  2-00000216  CHAR MS780C_RAM_TYPE                             104  (0:3)
  2-000002E6  CHAR MS780E_INTERLEAVE_MODE                      155  (0:4)
  2-0000027E  CHAR MS780E_RAM_TYPE                             104  (0:3)
  2-00000703  CHAR REF_MODE                                     44  (0:3)
  2-00000471  CHAR SBI_CONFIRM                                  36  (3)
  2-0000086B  CHAR SBI_FUNCTION                                299  (0:12)
  2-0000072F  CHAR SBI_RESPONSE                                 63  (0:2)
  2-00000790  CHAR SBI_TAG                                     144  (0:7)
  2-00000000  CHAR TIMEOUT_STATUS                               88  (0:3)
  2-0000059A  CHAR V1ACCS                                       20  (15:15)
  2-00000996  CHAR V1CI780_REGA                                 57  (8:10)
  2-00000426  CHAR V1DRCR                                  75  (3)
  2-00000112  CHAR V1MS780C_REGA                           17  (0:0)
  2-00000112  CHAR V1MS780E_REGA                           17  (8:8)
  2-000006B2  CHAR V1SBI_COMPARATR                         81  (29:31)
  2-00000495  CHAR V1SBI_ERROR                             66  (3)
  2-000005F9  CHAR V1SBI_FAULT                             92  (16:19)
  2-00000655  CHAR V1SBI_REGA                              93  (21:23)
  2-0000076E  CHAR V1SBI_SILO                              34  (30:31)
  2-00000550  CHAR V1TIMEOUT_ADDR                          29  (29:29)
  2-00000123  CHAR V1UBA_REGA                              63  (16:18)
  2-000005AE  CHAR V2ACCS                                  69  (27:29)
  2-000009CF  CHAR V2CI780_REGA                           125  (16:20)
  2-00000381  CHAR V2DRCR                                  24  (11:11)
  2-00000162  CHAR V2MS780E_REGA                          180  (15:20)
  2-000004D7  CHAR V2SBI_ERROR                             46  (7:8)
  2-00000058  CHAR V2SBI_FAULT                            186  (26:31)
  2-00000058  CHAR V2SBI_REGA                             186  (26:31)
  2-000005F3  CHAR V3ACCS                                   6  (31:31)
  2-00000399  CHAR V3DRCR                                 126  (15:20)
  2-00000505  CHAR V3SBI_ERROR                             75  (13:15)
  2-00000417  CHAR V4DRCR                                  15  (24:24)

LABELS

   Address    Label       Address    Label       Address    Label       Address    Label       Address    Label       Address    Label

 1-000000CE   10'      1-000000F6   30'      1-00000108   40'      1-0000011A   50'      1-00000135   60'      1-00000151   70'
 1-00000175   80'      1-0000018A   90'      1-0000019C   100'     1-000001B3   105'     1-000001D2   135'     1-000001E4   140'
 1-0000020F   150'     1-00000235   155'     1-00000247   160'     1-00000269   170'     1-00000293   175'     1-0000029C   180'
     **       183      1-000002B4   185'     1-000002D1   187'     1-000002EA   189'     1-00000304   191'     1-00000315   193'
 1-00000329   197'     1-0000033A   198'     1-0000034A   200'     1-00000363   209'     1-00000378   210'     1-00000390   215'
 1-000003A7   220'         **       230      1-000003D1   240'     1-000003E3   250'     1-00000412   255'     1-00000406   256'
 1-00000434   260'     1-00000454   261'     1-00000466   262'     1-0000046D   263'     1-00000479   264'     1-00000485   265'
 1-000004A5   269'     1-000004BA   275'     1-000004DF   277'     1-0000051B   279'     1-00000531   280'     1-00000548   285'
 1-0000055E   289'     1-00000574   295'     1-0000058B   300'     1-000005A8   400'     1-000005BB   405'

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name            Type  Name            Type  Name            Type  Name            Type  Name

   I*4  COMPRESS4        I*4  COMPRESSC        I*4  LIB$EXTZV             LINCHK                OUTPUT

COMMAND QUALIFIERS

  FORTRAN /LIS=LIS$:VAX780REG/OBJ=OBJ$:VAX780REG MSRC$:VAX780REG

  /CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /STANDARD=(NOSYNTAX,NOSOURCE_FORM)
  /SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)
  /F77  /NOG_FLOATING  /I4  /OPTIMIZE  /WARNINGS  /NOD_LINES  /NOCROSS_REFERENCE  /NOMACHINE_CODE  /CONTINUATIONS=19
```

```
COMPILATION STATISTICS

  Run Time:           13.57 seconds
  Elapsed Time:       29.06 seconds
  Page Faults:        278
  Dynamic Memory:     290 pages
```

XXX
V04

0155 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

EVLDEF
MDL

LIBTAIL
B32

XXXQUE
LIS

UTILDEF
REQ

ERRDEF
MDL

EVCDEF
MDL

CONSOLE
LIS

USEROPEN
LIS

VAXPSL
LIS

ERRFMT

VAX780REG
LIS

ERRFMT
MAP

EVLIBRARY
B32

VAX750REG
LIS

VECMAPREG
LIS

ERRFMT
LIS

EVL

LIBHEAD
B32

EVL
MAP

WQDEF
MDL